Section IV:

AMENDMENT UNDER 37 CFR §1.121

REMARKS

Summary of Telephone Interview

On August 13, 2003, the examiner and applicant's agent held a telephone interview at the applicant's agent's request in order to clarify differences between the cited art and the claimed invention, and to establish terms of mutually agreeable of definitions and scope with reference to a proposed amendment.

Applicant's agent pointed out that the cited references did not utilize a system dictionary as a control list of format tags to be replaced, but instead used other methods of generating layout templates. Examiner stated that she was unclear as to the confusion with the definition of "dictionary" as used in the claims, e.g. whether it was a traditional language-to-language type of dictionary such as an English-to-French translator's dictionary, or not.

Applicant's agent directed the examiner's attention to the disclosure at page 21, lines 4 - 6, and page 23, lines 14 - 19. In this portion of the text, the preferred embodiment makes "entries" into a "system dictionary", such as the ***java.util.Dictionary***, for each HTML TAG to be processed to swap the existing content with new or alternate content. Figure 6 shows the step of creating these entries, while Figures 7 and 8 show the steps of searching the source or original HTML for layout tags which are also in the dictionary, and replacing the content parameters for those tags with alternate content.

Examiner then agreed that the cited art did not utilize such a dictionary, and that she was unaware of any systems which did use a dictionary as such. Further discussion was had regarding language in the claims which might be adopted to make this clarification. Additionally, some discussion was held regarding how such entries were to be made, and what they may look like.

Examiner requested that applicant's response include such a clarification of the definition of the term "system dictionary", and clarification of how such entries are made in a such a dictionary. Applicant's agent agreed to provide this information in the reply to the Office Action.

Objections to the Figures

In the previous Office Action, the draftsperson has objected to Figures 2 and 3 for reasons of unacceptable top margins. Examiner has indicated that formal corrected drawings may be filed at

allowance.

Rejections under 35 U.S.C. §112

In the Office Action, examiner has rejected claims 22 - 25 under 35 U.S.C. §112 as lacking antecedent basis for the term "static web page" in lines 3 - 4. These claims have been amended in this reply to recite "web document portion", which is supported by the claims upon which they depend.

Rejections under 35 U.S.C. §103

In the Office Action, examiner has rejected all claims under 35 U.S.C. §103(a) over US Patent 6,356,903 to Baxter in view of US Patent 6,559,861 to Kennelly. However, neither Baxter nor Kennelly teach use of a system dictionary as a control list of layout tags to be processed with alternate content, as discussed during our telephone interview.

To substantiate our definition of "system dictionary" as disclosed on page 21 of our specification, we are attaching to this reply a copy of page 227 of "Training Guide: Java [TM] 2 Certification" by Jamie Jaworski, from New Riders Publishing, Copyright 1999. This page describes the well-known Java utilities package Dictionary class as a set of abstract functions for handling key-value pairs, including a "put" function which can be used to create a new key-value pair. Our disclosure includes an example "put" statement on page 21.

This preferred embodiment is of course not the only way to create a system dictionary to perform the function of controlling which layout tags are processed, so we have claimed the use of a "system dictionary" with "entries" indicating which layout tags are to be processed. Figures 6, 7, and 8 support the creating and use of this dictionary for this controlling function, as well, and our disclosure at page 23, lines 14 - 19 specifically describes the search of the HTML for tags which are in the dictionary.

As such, Baxter in view of Kennelly does not preclude patentability of our claims as amended, and we request reconsideration and allowance of the claims as amended.

1. The combination or modification of the references in the manner suggested by the examiner does not teach all the claimed elements, steps, or restrictions. MPEP §2143.03 states:

All Claim Limitations Must Be Taught or Suggested. To establish *prima facie*

obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art.

The facts derived from the references and set forth below indicate that the examiner's suggested combination and modification of the cited references does not teach all claimed elements, limitations or step. Therefore, the rejection is unsupported by the art and should be withdrawn.

i. Baxter in view of Kennelly fails to teach or suggest providing one or more entries in a system dictionary, each entry representing a static layout tag type to be mapped to one or more dynamic structures (first element of claims 1, 18 and 33); and

ii. Baxter in view of Kennelly fails to teach or suggest mapping alternate web content into an extracted layout template web page by replacing content parameters associated with said extracted layout definitions for which matching entries are found in said system dictionary, thereby creating at least one web page portion containing said alternate (last element of claims 1, 18, and 33).

Conclusion

It has been established that the rejections are not supported by the cited art, especially in view of the comments presented herein, the widely-accepted terminology as indicated by contemporary text books, and the present amendments to the claims. Reconsideration of all rejections is hereby requested.

###

Page 227 of "Training Guide: Java 2 Certification"

The BitSet access methods provide and, or, and exclusive or logical operations on bit sets, enable specific bits to be set and cleared, and override general methods declared for the Object class.

The Dictionary Class

The Dictionary, Hashtable, and Properties classes are three generations of classes that implement the capability to provide key-based data storage and retrieval. The Dictionary class is the abstract superclass of Hashtable, which is, in turn, the superclass of Properties.

Dictionary provides the abstract functions used to store and retrieve objects by key-value associations. The class allows any object to be used as a key or value. This provides great flexibility in the design of key-based storage and retrieval classes. Hashtable and Properties are two examples of these classes.

The Dictionary class can be understood using its namesake. A hard-copy dictionary maps words to their definitions. The words can be considered the keys of the dictionary, and the definitions are the values of the keys. Java dictionaries operate in the same fashion. One object is used as the key to access another object. This abstraction will become clearer as you investigate the Hashtable and Properties classes.

The Dictionary class defines several methods that are inherited by its subclasses. The elements() method is used to return an Enumeration object containing the values of the key-value pairs stored within the dictionary. The keys() method returns an enumeration of the dictionary keys. The get() method is used to retrieve an object from the dictionary based on its key. The put() method puts a Value object in the dictionary and indexes it using a Key object. The isEmpty() method determines whether a dictionary contains any elements, and the size() method identifies the dictionary's size in terms of the number of elements it contains. The remove() method deletes a key-value pair from the dictionary, based on the object's key.

NOTE

Dictionary Is Obsolete The Dictionary class has been rendered obsolete by the Map interface, as of JDK 1.2. However, its Hashtable and Properties subclasses are still in use.

The Hashtable Class

The Hashtable class implements a hash table data structure. A *hash table* indexes and stores objects in a dictionary using hash codes as the objects' keys. *Hash codes* are integer values that identify objects. They are computed in such a manner that different objects are very